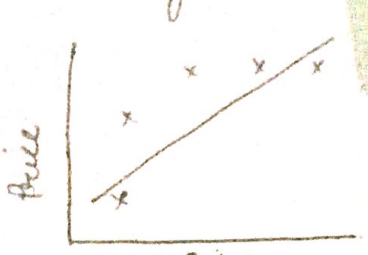
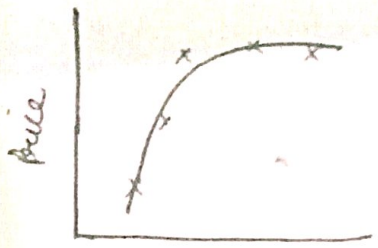
7.1- Regularisation | the Problem of overfitting



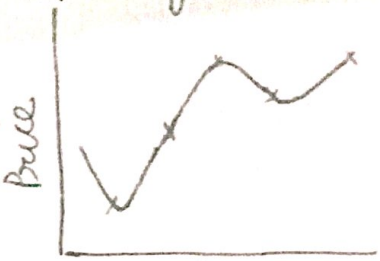
$\rightarrow \theta_0 + \theta_1 x$

Underfit - High bias



$\theta_0 + \theta_1 x + \theta_2 x^2$

Just right

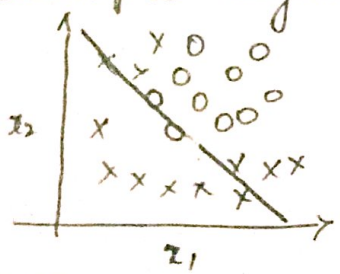


$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Overfit High variance

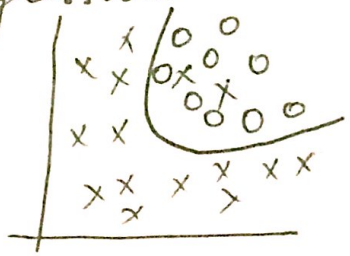
Overfitting - If we have too many features, that learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fails to generalise to new examples (predict price on new examples).

Example: Logistic regression

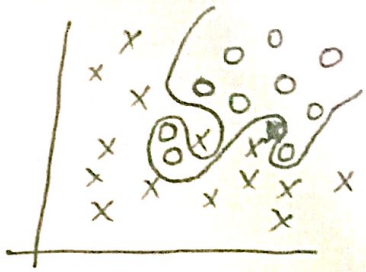


$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2) + \theta_5 x_2$



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \cdots)$

overfit

Addressing overfitting:

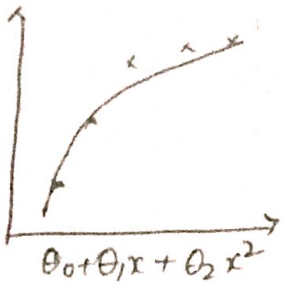
→ If features are very great and training example are less then it leads to overfitting

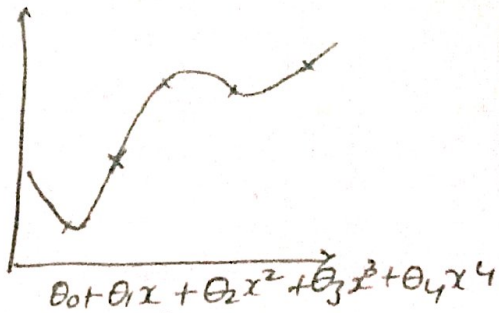
Option: 1. Reduce number of features
- Manually select which features to keep
- Model selection algorithm (later in course)

2. Regularisation
- Keep all the features, but reduce magnitude/values of parameter θ_j
- Works well when we have a lot of features, each of which contributes a bit to predicting y.

7.2 Regularisation of cost function

Intuition
- Suppose we penalize and make θ_3, θ_4 really small





- $\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\theta_3^2 + 1000\theta_4^2$

$\theta_3 \approx 0$, $\theta_4 \approx 0$

Small values for parameters $\theta_0, \theta_1, \dots \theta_n$
- Simpler hypothesis
- Less prone to overfitting

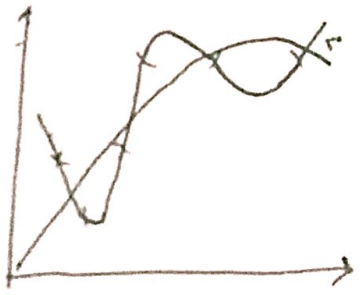
Housing
- Features: $x_1, x_2, \dots, x_{100}$
- Parameters $\theta_0, \theta_1, \theta_2, \dots \theta_{100}$

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

(λ: regularisation parameter)

λ - ensures tradeoff between fitting the data and keeping parameter small.

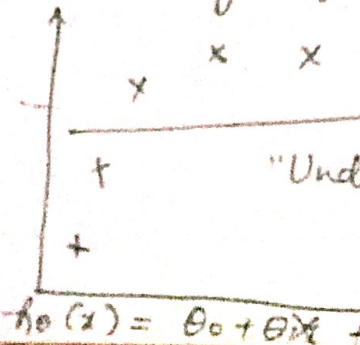
$\min_\theta J(\theta)$



In regularized linear regression, we choose θ to minimize

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{i=1}^{n} \theta_j^2\right]$$

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?



$\theta_1, \theta_2, \theta_3, \theta_4$

$\theta_1 \approx 0$, $\theta_2 \approx 0$

$\theta_3 \approx 0$, $\theta_4 \approx 0$

$h_\theta(x) = \theta_0$

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

7.3 - Regularization | Regularized Linear Regression

Gradient descent

Repeat {

$\theta_0 := \theta_0 - \alpha \boxed{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}}$ $\quad \frac{\partial}{\partial \theta_0} J(\theta)$

$\theta_j := \theta_j - \alpha \boxed{\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} - \frac{\lambda}{m}\theta_j\right]}$ $\quad \frac{\partial}{\partial \theta_j} J(\theta)$

}

$$\theta_j := \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}x_j^{(i)})$$

Normal Equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$\rightarrow \min_\theta J(\theta)$ $\qquad \frac{\partial}{\partial \theta_j} J(\theta) \overset{\text{set}}{=} 0$

$$\theta = \left(X^TX + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}\right)^{-1} X^Ty$$

$(n+1)\times(n+1)$

Non-invertibility (optional)

Suppose $m \leq n$

(# examples) (# features)

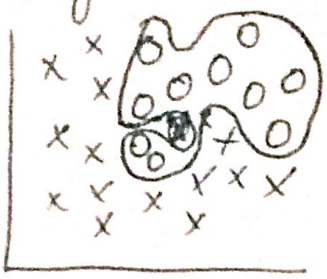
$\theta = (X^TX)^{-1}X^Ty$

↖ non-invertible | singular

If $\lambda > 0$

$$\theta = \underbrace{\left(X^TX + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}\right)^{-1}}_{\text{invertible}} X^Ty$$

7.4- Regularised Logistic Regression



$$h_\theta(x) = g(\theta_0 + \theta_1 x + \theta_2 x_1^2 + \theta_3 x_1^2 x_1 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^3 x_2^2 + \dots)$$

cost function :

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

Gradient descent

Repeat {

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} - \frac{\lambda}{m}\theta_j\right]$$

$(j = 1, 2, 3, \dots n)$

Advanced optimisation

```
function [jval, gradient] = costFunction(theta)
  jval = [code to compute J(θ)]
  gradient(1) = [code to compute ∂/∂θ0 J(θ)];
  gradient(2) = [code to compute ∂/∂θ1 J(θ)];
  .
  .
  gradient(n+1) = [code to compute ∂/∂θn J(θ)];
```

15-1 Anomaly Detection Problem

Aircraft engine features:
x_1 = heat generated
x_2 = vibration intensity

Dataset: $\{x_1, x_2, \ldots x_m^{(m)}\}$

New engine: x_{test}

Density estimation
Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$
Is x_{test} anamalous?

Anomaly detection example:
→ Fraud detection:
$x^{(i)}$ = features of user i's activities
Model $p(x)$ from data
Identify unusual users by checking which have $p(x) < \varepsilon$
→ Manufacturing
→ Monitoring computers in a data center.
$x^{(i)}$ = features of machine i
x_1 = memory use, x_2 = number of disk access/sec,
x_3 = CPU load, x_4 = CPU load / network traffic

15-2 Gaussian Distribution

Gaussian (Normal) distribution
Say $x \in \mathbb{R}$ If x is a distributed Gaussian with mean μ, variance σ^2

$$p(x, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

15-3 Anomaly detection algorithm

Training set: $\{x^{(1)}, x^{(2)}, \ldots x^{(m)}\}$
Each example is $x \in \mathbb{R}^n$

$x_1 \sim N(\mu_1, \sigma_1^2)$
$x_2 \sim N(\mu_2, \sigma_2^2)$
$x_3 \sim N(\mu_3, \sigma_3^2)$

$p(x)$
$= p(x_1; \mu_1, \sigma_1^2)\, p(x_2; \mu_2, \sigma_2^2)\, p(x_3; \mu_3, \sigma_3^2)$
$\cdots\, p(x_n; \mu_n, \sigma_n^2)$
$= \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$

1. Choose features x_i that you think might be indicative of anamalous examples
2. Fit parameter $\mu_1, \mu_2, \ldots \mu_n, \sigma_1^2, \ldots, \sigma_2^2$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2$$

3. Given new examples x, compute $p(x)$:

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

Anomaly if $p(x) < \varepsilon$

15.4 - Developing and Evaluating an anamaly Detection System

The importance of real-number evaluation

When developing a learning algorithm (choosing features, etc), making decisions is much easier if we have a way of evaluating our learning algorithm

Assume we have some labeled data, of anomalous and non-anamalous examples. ($y = 0$ if normal, $y = 1$ if anamalous)

→ Training set: $x^{(1)}, x^{(2)}, \ldots x^{(m)}$ (assume normal examples / not anamalous)

→ Cross validation set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \ldots (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$

→ Test set: $(x_{test}^{(1)}, y_{test}^{(1)}) \ldots (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

Aircraft engine motivating examples:

10000 good (normal) engines

20 flawed engines (anamalous)

Training set: 6000 good engines

CV: 2000 good engines ($y=0$), 10 anamalous ($y=1$)

Test: 2000 good engines ($y=0$), 10 anamalous ($y=1$)

Alternative:

Training set: 6000 good engines

CV: 4000 good engines ($y=0$), 10 anomalous ($y=1$)

Test: 4000 good engines ($y=0$), 10 anomalous ($y=1$)

Algorithm evaluation

Fit model $p(x)$ on training set $\{x^{(1)}, \ldots, x^{(m)}\}$

On a cross-validation / test example x, predict

$$y = \begin{cases} 1 & \text{if } p(x) < \varepsilon \text{ (anomaly)} \\ 0 & \text{if } p(x) \geq \varepsilon \text{ (normal)} \end{cases}$$

Possible evaluation metrics:

- True positive, false positive, false negative, true negative
- Precision / Recall
- F1-score

Can also use cross validation set to choose parameter ε

15.5 - Anomaly Detection Vs Supervised Learning

Anomaly detection	Supervised learning
Very small number of positive example ($y=1$). (0-20 is common)	Large number of positive and negative examples
Large number of negative ($y=0$) examples	Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set
Many different 'types' of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far	

Application

Anamaly detection

Fraud detection

Manufacturing (eg. aircraft engines)

Monitoring machines in a data center

Supervised Learning

Email spam classification

Weather prediction (sunny/rainy/etc).

Cancer classification

15.6 - Choosing the Features to Use

Non-gaussian features

→ Make it gaussian by taking log, $x^{\frac{1}{2}}$, $x^{\frac{1}{3}}$ - - -

Error analysis for anamaly detection

Want $p(x)$ large for normal example x

$p(x)$ small for anamalous example x

Most common problem:

$p(x)$ is comparable (say, both large) for normal and anamalous examples

Monitors computers in a data center

Choose features that might take on unusually large or small values in the event of an anamaly

x_1 = memory use of computer

x_2 = number of disk accesses/sec

x_3 = CPU load

x_4 = network traffic

15.7 - Multivariate Gaussian Distribution

$x \in \mathbb{R}^n$. Don't model $p(x_1)$, $p(x_2)$, . . ., etc separately

Model $p(x)$ all in one go

Parameters: $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$ (covariance matrix)

15.8 - Anamaly Detection using multivariate Gaussian Distribution

Parameter μ, Σ

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

Parameter fitting:

Given training set: $\{x^{(1)}, x^{(2)}, \ldots x^{(m)}\}$

$$\mu = \frac{1}{m} \sum x^{(i)} \qquad \Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

1. Fit model $p(x)$ by setting

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

2. Given a new example x, compute

$$p(x) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

Flag an anamoly if $p(x) < \varepsilon$

Relationalship to original model

Original model: $p(x) = p(x_1; \mu_1, \sigma_1^2) \times p(x_2; \mu_2, \sigma_2^2) \times \cdots \times p(x_n, x_n, \sigma_n^2)$

Corresponding to multivariate Gaussian

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

Original model

$p(x_1; \mu_1, \sigma_1^2) \times \cdots \times p(x_n; \mu_n, \sigma_n^2)$

Manually creates features to capture anamalies where x_1, x_2 take usual combinations of values

Computational cheaper (alternatively scales better to large n)

OK even if m (training set size) is small

Gaussian model

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

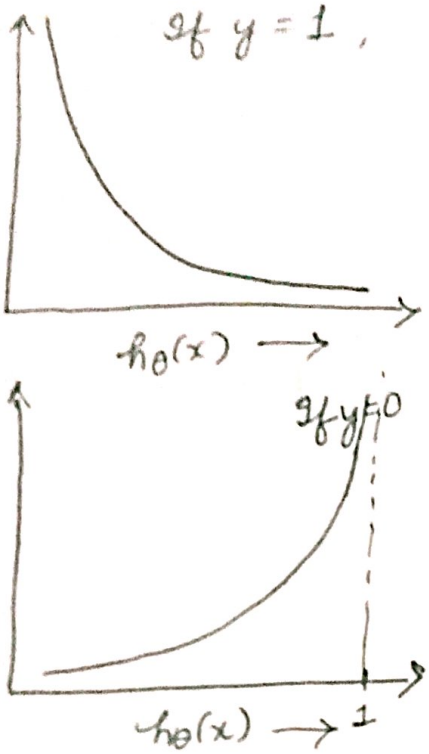
Automatically captures correlations between features

Computational more expensive

Must have $m > n$ or else Σ is non-invertible

6.4 Logistic regression cost function

$$\text{cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



cost = 0 if $y = 1$, $h_\theta(x) = 1$

But as $h_\theta(x) \to 0$

Cost → ∞

capture intuition that if $h_\theta(x) = 0$, (predict $P(y=1 \mid x; \theta) = 0$), but $y = 1$, we will penalise learning algorithm by a very large cost.

6.5 - Simplified Cost Function and Gradient Descent

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

$$\text{cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1-y) \log(1 - h_\theta(x))$$

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

To fit parameter θ:

$$\min_\theta J(\theta)$$

To make a prediction given new x:

Output $h_\theta(x) = \dfrac{1}{1 + e^{-\theta^T x}}$

Repeat {

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneous update all θ_j)

}

Algorithm look similar to logistic regression

6.6 — Advanced Optimisation

Optimisation algorithm

Cost function $J(\theta)$. Want $\min_\theta J(\theta)$

Given θ, we have code that can compute

$J(\theta)$

$\frac{\partial}{\partial \theta_j} J(\theta)$ (for $j = 0, 1, 2, \ldots n$)

Gradient descent:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

Optimisation algorithms: Gradient Descent
Conjugate gradient
BFGS
L-BFGS

Example: $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ $\min_\theta J(\theta)$ $\theta_1 = 5, \theta_2 = 5$

$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$

$\frac{\partial}{\partial \theta} J(\theta) = 2(\theta_1 - 5)$

$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$

Advantage: No need to manually pick α
Often faster than gradient descent

Disadvantage: - More complex

6.7 - Multiclass Classification

One-Vs-all (one-vs-rest)

$h_\theta^{(i)}(x) = P(y = i | x; \theta)$ $(i = 1, 2, 3)$

Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$

On a new input x, to make a prediction, pick the class i that maximises $\max_i h_\theta^{(i)}(x)$